**Round Robin Scheduling**

**#include <iostream>**

**#include <algorithm>**

**#include <iomanip>**

**#include <queue>**

**#include <cstring>**

**using namespace std;**

**struct process {**

**int pid;**

**int arrival\_time;**

**int burst\_time;**

**int start\_time;**

**int completion\_time;**

**int turnaround\_time;**

**int waiting\_time;**

**int response\_time;**

**};**

**bool compare1(process p1, process p2)**

**{**

**return p1.arrival\_time < p2.arrival\_time;**

**}**

**bool compare2(process p1, process p2)**

**{**

**return p1.pid < p2.pid;**

**}**

**int main() {**

**int n;**

**cout<<"Enter the number of processes: ";**

**cin>>n;**

**int tq;**

**struct process p[n];**

**float avg\_turnaround\_time;**

**float avg\_waiting\_time;**

**int total\_turnaround\_time = 0;**

**int total\_waiting\_time = 0;**

**int total\_response\_time = 0;**

**int total\_idle\_time = 0;**

**int burst\_remaining[n];**

**int idx;**

**cout << setprecision(2) << fixed;**

**cout<<"Enter time quantum: ";**

**cin>>tq;**

**cout<<"Enter arrival time of process ";**

**for(int i=0;i<n;i++)**

**{**

**cin>>p[i].arrival\_time;**

**}**

**cout<<endl;**

**cout<<"Enter burst time of process ";**

**for(int i=0;i<n;i++)**

**{**

**cin>>p[i].burst\_time;**

**burst\_remaining[i] = p[i].burst\_time;**

**p[i].pid = i+1;**

**}**

**cout<<endl;**

**sort(p,p+n,compare1);**

**queue<int> q;**

**int current\_time = 0;**

**q.push(0);**

**int completed = 0;**

**int mark[100];**

**memset(mark,0,sizeof(mark));**

**mark[0] = 1;**

**while(completed != n) {**

**idx = q.front();**

**q.pop();**

**if(burst\_remaining[idx] == p[idx].burst\_time) {**

**p[idx].start\_time = max(current\_time,p[idx].arrival\_time);**

**total\_idle\_time += p[idx].start\_time - current\_time;**

**current\_time = p[idx].start\_time;**

**}**

**if(burst\_remaining[idx]-tq > 0) {**

**burst\_remaining[idx] -= tq;**

**current\_time += tq;**

**}**

**else {**

**current\_time += burst\_remaining[idx];**

**burst\_remaining[idx] = 0;**

**completed++;**

**p[idx].completion\_time = current\_time;**

**p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;**

**p[idx].waiting\_time = p[idx].turnaround\_time - p[idx].burst\_time;**

**p[idx].response\_time = p[idx].start\_time - p[idx].arrival\_time;**

**total\_turnaround\_time += p[idx].turnaround\_time;**

**total\_waiting\_time += p[idx].waiting\_time;**

**total\_response\_time += p[idx].response\_time;**

**}**

**for(int i = 1; i < n; i++){**

**if(burst\_remaining[i] > 0 && p[i].arrival\_time <= current\_time && mark[i] == 0) {**

**q.push(i);**

**mark[i] = 1;**

**}**

**}**

**if(burst\_remaining[idx] > 0) {**

**q.push(idx);**

**}**

**if(q.empty()) {**

**for(int i = 1; i < n; i++) {**

**if(burst\_remaining[i] > 0) {**

**q.push(i);**

**mark[i] = 1;**

**break;**

**}**

**}**

**}**

**}**

**avg\_turnaround\_time = (float) total\_turnaround\_time / n;**

**avg\_waiting\_time = (float) total\_waiting\_time / n;**

**sort(p,p+n,compare2);**

**cout<<endl;**

**cout<<"Process\t"<<"\tArrivalTime\t"<<"BurstTime\t"<<"CompletionTime\t"<<"TurnAroundTime\t"<<"WaitingTime\t"<<"ResponseTime\t"<<"\n"<<endl;**

**for(int i = 0; i < n; i++) {**

**cout<<p[i].pid<<"\t\t"<<p[i].arrival\_time<<"\t\t"<<p[i].burst\_time<<"\t\t"<<p[i].completion\_time<<"\t\t"<<p[i].turnaround\_time<<"\t\t"<<p[i].waiting\_time<<"\t\t"<<p[i].response\_time<<"\t\t"<<"\n"<<endl;**

**}**

**cout<<"Average Turnaround Time = "<<avg\_turnaround\_time<<endl;**

**cout<<"Average Waiting Time = "<<avg\_waiting\_time<<endl;**

**}**